Supporting FIPA Interoperability for Legacy Multi-Agent Systems

Christos Georgousopoulos[1], Omer F. Rana[1], and Anthony Karageorgos[2]

[1] Department of Computer Science, Cardiff University, P.O.Box 916,
Cardiff CF24 3XF, UK
{geolos, o.f.rana}@cs.cf.ac.uk
http://www.cs.cf.ac.uk/Digital-Library
[2] Department of Computation, UMIST, Manchester, M60 1QD, UK
karageorgos@co.umist.ac.uk

Abstract. A system that conforms to FIPA specifications (standards) is a FIPA-compliant system, and can interoperate with any other heterogeneous systems which are FIPA-compliant also. The conversion of a Multi-Agent System (MAS) into a FIPA-compliant system (i.e. one that adheres to FIPA standards), is important to support interoperability across different MAS. A different approach to conforming a MAS to a FIPA-compliant one, other that the common one of converting the whole system to adhere to FIPA specifications, is the use of the FIPA-compliant gateways[7]. In this paper we extent our work on the FIPA-compliant gateways and we demonstrate the successful interoperability of the gateways based on a MAS utilising an active digital library composed of multi-spectral images of the Earth, as part of the Synthetic Aperture Radar Atlas (SARA).

1 Introduction

The conversion of a Multi-Agent System (MAS) into a FIPA-compliant system (i.e. a system that adheres to FIPA standards), implies that system developers must rebuild their systems based on FIPA specifications. Such a conversion imposes amendments on the system architecture to conform to the new standards, which may results in extensive code rewriting and testing.

We extend our work on the FIPA-compliant gateways[7] here. We describe how a developer may attach FIPA-compliant gateways to a MAS and briefly discuss the advantages of adopting this approach. Note that our approach should not be confused with agent software integration support for FIPA specifications, or with similar approaches that claim FIPA compliance but they actually alter[12] the original FIPA specifications. Finally, we demonstrate how interoperability can be applied to a MAS, with particular emphasis on the SARA (Synthetic Aperture Radar Atlas) architecture[15] by providing results of our experiments.

The approach presented in this paper is particular relevant to Agent-oriented software engineering as it enables existing agent systems to be integrated in a seamless manner. When engineering agent systems, it is likely that different implementation and design approaches have been adopted. The "gateway" approach presented here may also be used to combine such systems together, and enable interaction between them using FIPA-based performatives. Although we are aware that standards are likely to be modified over time, FIPA provides the most valuable agent interoperability specification at the present time. Our approach is therefore focused on supporting interaction between agent systems that adhere to this standard.

2 Building FIPA-Compliant Gateways

Based on the guidelines provided by the FIPA association, for an agent platform implementation to be considered FIPA-compliant it must at least implement the "Agent Management" and "Agent Communication Language" specifications, which should conform to the latest *experimental* and/or *standard* status specifications.

The usual approach to conforming a MAS into a FIPA-compliant one is to modify the whole system based on FIPA specifications. A different approach that has not yet been adopted by any developer is to amend just a part of the system's architecture. The top picture of figure 1, represents a typical multi-agent system (MAS 1) that has been conformed to FIPA specifications in order to be able to interoperate i.e. receive/send data from/to other FIPA-compliant multi-agent systems (EXternal MAS). Figure 1b, represents our approach to conforming a MAS into a FIPA-compliant one. The actual architecture of the system remains the same as before, but two FIPA-compliant gateways (in grey) have to be added to the system. These work as *adaptors (wrappers)* to ensure interoperability with other FIPA-compliant external multi-agent systems (EX MAS). Interoperability in this sense applies at both the communication and application levels. The communication level comprises the connection and communication layer, whereas the application level comprises the ontological and agent service layer[4].

The two *gateways* are the FIPA-compliant part of the system. Each of those has all of the mandatory, normative components of the FIPA architecture. The use of these FIPA-compliant gateways is depicted in figure 5 of section 4.2, where we demonstrate the adoption of the FIPA-compliant gateways by the SARA multi-agent system. Each *gateway* contains three agents: the Agent Management System (AMS), the Directory Facilitator (DF) and the *gateway* agent. The AMS and DF are the FIPA agents, as defined by FIPA specifications. The *gateway* agent is the only agent of the system registered by both AMS and DF, which acts as a wrapper between MAS2 and any external MAS. All the available services of the system are represented by this agent. It is like having an ordinary FIPA compliant system with only one registered agent capable of providing services. The Directory Facilitator (DF) and Agent Communication Channel (ACC), support the required infrastructure for enabling service interoperability, and are part of the

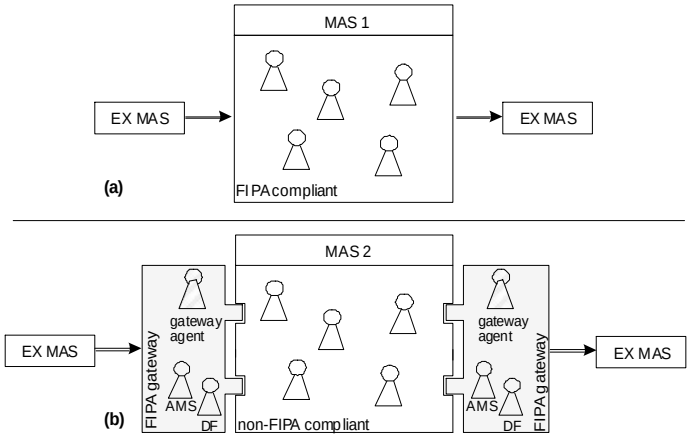

Fig. 1. Two different approaches of conforming an agent platform into a FIPA-compliant one

FIPA specifications. The communication between an EX MAS and MAS2 is accomplished though the Agent Communication Channel (ACC) and the protocols that are supported (concerning the connection layer) are reflected thought the platform address. The *gateway* agent communicates with agents from EX MAS using the FIPA Agent Communication Language (ACL). Its responsibility is to translate the incoming messages to a form understood by its internal agents i.e. the agents that are hidden by the EX MAS. Likewise, the internal agents' requests have to be also converted by the *gateway* agent into ACL messages, in order to be understood by an EX MAS. The *gateway* agent maintains a list of the agents within the system being wrapped, along with the registered services (with DF) that each of them can provide. Therefore, based on the service requested by an EX MAS, the *gateway* agent knows to which system agent the message should be forwarded, after it has been translated into the form understood by the appropriate agent that receives the request.

Hence, the external MAS does not *see* anything else apart from the *gateway* agent; which on receiving a request from an external MAS (on the left side of MAS2) is responsible for transferring the request to the agents of its system, which are hidden by the external MAS, for processing the request. Once the request is accomplished, a response is returned to the external MAS through the *gateway* agent. In the case where agents from MAS 2 need to communicate with an external MAS (on the right side of MAS2), their request is passed through the *gateway* agent and translated into ACL; the results gathered by the external MAS are returned to MAS 2 agents through the *gateway* agent as well.

The gateway agent also supports agent conversation sessions by supplying the conversation ID (of its communication with the external agent) to its appropriate internal agent along with the translated message. Once, it receives feedback from one of its internal agents, it replies to the corresponding external agent on the conversation indicated

by the conversation ID received by the former one i.e. the conversation ID that the gateway agent had initially sent to its internal agent.

The capability of the FIPA-compliant gateway may be further extended by defining extra sets of operations that may be supported by these agents. For instance, the utilisation of a security layer will enable heterogeneous MAS to interoperate using X509 based digital certificates. In addition, an agent mobility layer would provide the capability to support agent migration between heterogeneous MAS built on the same agent platform.

2.1 Supporting Multiple Gateway Agents

Although one of the advantages of the FIPA-compliant gateway is to *isolate* the externally accessible part of the architecture i.e. the gateways, from the rest of the system for increasing security (since the policy of the architecture remains hidden to a foreign Agency), some developers might need to *expose* more than one agents to an external MAS.

This could be achieved by adding multiple gateway agents to the FIPA-compliant gateway that provides interoperability between the legacy MAS and an external one, as shown in figure 2a. In this case, the agent that would need to be directly accessed by an external MAS, and could be represented by a separate gateway agent. For instance, with reference to figure 2, agent1 with service1 is resented by gateway agent1 (GA), service2 of agent2 by GA2 and service3/4 & 5 by GA3.

Even in the case where all of the available services provided by a legacy MAS are represented by a single gateway agent, the introduction of multiple gateway agents with replicated services in the FIPA-compliant gateway may also be useful for:

– Balancing the incoming requests among the existing gateway agents. In a MAS with numerous received requests, the gateway agent that receives a request from an EX MAS may pass the request to another (less occupied) gateway agent. For instance, the steps that have to be followed in order for a message to be passed from one gateway agent to another one, see figure 2b, are:

  Step 1: An agent from an EX MAS sends a request to GA1.

  Step 2: If the message is not understood by GA1, it replies to the sender agent with a Not-understood message, otherwise it sends an Agree message including the parameter reply-to with the gateway agent's name to which the message is forwarded i.e. GA2. Therefore, subsequent messages (from the external agent) will be directed to GA2.

  Step 3: GA1 forwards the external agent's message to GA2 via an *Inform* message including the parameter *reply-to* with the external agent's name.

  Step 4: GA2 communicates with its appropriate internal agent according to the service required. The message that is sent to the internal agent is the content of the GA1's message, which has already been translated by GA1 (to validate the external agent's message) to the form understood by their internal agents.

Step 5: GA2 upon receipt of results from its internal agent, generates an ACL message and sends it to the external agent via an *Inform* message.

– Increasing fault tolerance of the interoperability part of a legacy MAS. The FIPA-compliant gateways may be configured to be distributed i.e. each gateway agent to be distributed on a different host. Therefore, even if one of the gateway agents fails, the MAS may still be able to provide its services to an external MAS through the rest of the gateway agents.

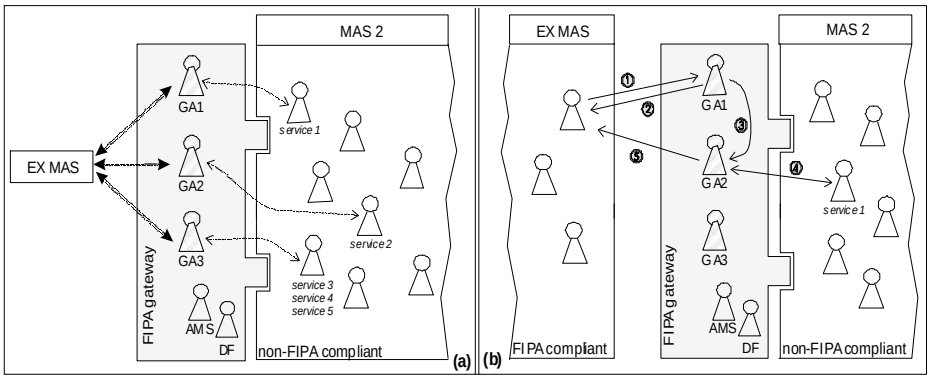

Fig. 2. Multiple gateway agents

To conclude, there are three case scenarios for the FIPA-compliant gateway that provides interoperability between the legacy MAS and an external one: (a) a single gateway agent with all the available services registered under its entity (b) a gateway agent per service (c) multiple gateway agents with replicated services. According to the MAS that need to address FIPA interoperability, developers can choose one of the above scenarios that suit their needs.

2.2 Advantages of FIPA-Compliant Gateways

The alternative approach of using FIPA-compliant gateways[7] for conforming a legacy MAS into a FIPA-compliant, yields the following advantages:

– System's architecture remains the same as before. Implementation is only needed for the FIPA-compliant gateways and the interaction between the gateway agents with the other agents of the system. The continuous improvement of FIPA specifications have a direct affect on the developer's systems since they should conform to the latest specifications. Consequently, developers can save time in terms of design and implementation by applying the new standards (the FIPA revised specifications) only

in a specific part of their system i.e the FIPA-compliant gateways; avoiding the complexity of amending the whole system.

- Security is increased. There is still no coherent agent security details from FIPA at this time. Although, FIPA is planning in the future to investigate security related issues within FIPA architecture, and formulate a long term strategy for the integration of security features into FIPA specifications[3], [10], there is currently debate as to whether a generic or default level of agent security ought to be specified. It is also required that such security criteria be applicable to different types of agent infrastructures and application domains[13]. Based on the gateways approach, *isolating* the interoperable part of the architecture (i.e. the gateways) from the rest of the system increases security. The policy of the architecture remains hidden to the foreign Agency due to the FIPA-compliant gateways. The interaction between the system and a foreign agency is managed by the *gateway* agent; the rest of the agents, hardware/software resources cannot be accessed. Securing the FIPA-compliant gateways, from where foreign malicious agents can enter into the system, implies minimum security for the rest of the system. Therefore, the FIPA-compliant gateways can also act as a shield for the core system. Requirements and design issues for adding security to FIPA agent systems can be found in [13].
- Performance is improved. FIPA specifications exist for the intercommunication between heterogeneous agent systems i.e agents that are hosted on different platforms. Consider an agent system which does not need to communicate with an external one. The conversion of such a system into a FIPA-compliant one would be useless, since the agents which belong to the system can obviously interoperate between themselves. Since interoperability can be achieved with the use of the FIPA-compliant gateways without actually affecting the actual system, it is unnecessary to conform the whole system to FIPA specifications. For instance, the existence of the Directory Falicitator (DF), Agent Management System (MAS), Agent Communication Channel (ACC) and Internal Platform Message Transport (IPMT), which are mandatory, normative components of the FIPA architecture, impose extra complexity and delay in a system constituted by homogeneous agents capable of interoperating between themselves. It is likely that a legacy agent system will not utilise the FIPA Agent Communication Language (ACL), especially if the agents within such a system are identical. An important role of the gateway, in this context, is to translate messages from a FIPA-compliant to a legacy system. There are also cases where agents exchange data in the form of simple Strings i.e. without the need of parsing and unparsing the transmitted information. In addition, the FIPA-compliant gateways have a direct affect on the security of the system and therefore on its performance. The more secure the FIPA-compliant gateways are, the less security is needed for the rest of the system. For instance, the cost of encrypting the messages transmitted between the agents, apart from the gateway agent, can be avoided. Consequently, the minimization of security (apart from the FIPA-compliant gateways) increases the overall performance of the system.

3 Steps of Deployment

The deployment of the FIPA-compliant gateways involves the following steps: (a) the creation and configuration of the two FIPA-compliant gateways i.e. one to support interoperability between an external MAS and the legacy one, and vice-versa, and (b) the creation of each of the gateway agents i.e. one per gateway.

3.1 Creation of the FIPA-Compliant Gateways

As mentioned in section 2, an agent platform implementation to be considered FIPA-compliant, it must at least adhere to the latest FIPA "Agent Management" and "Agent Communication Language" specifications. Therefore, the gateways should also adhere to those specifications.

The creation of the gateways, that will adhere to those specifications, may be easily achieved by using a toolkit like FIPA-OS[5]. After the initial installation of FIPA-OS only the configuration[6] of the platform remains. Briefly, this includes the configuration of the:

- platform profile: describes information about the FIPA-OS platform, including the platform's host-name, the 'location' of the AMS and location of other profiles used by entities within the platform (i.e. gateway agents, ACC). The identification of a Naming Service is also necessary for agents on a platform to locate one another.
- ACC profile: provides configuration information for the ACC of the platform, including the internal MTPs the platform is using, the external MTPs for communication with external MAS, details of other platforms that should be contacted at start-up.

Once the configuration is finished, the execution of a simple FIPA-OS script starts-up the configured FIPA-agent platform with the AMS and DF agents initialized. The last piece remaining for the implementation of the FIPA-compliant gateways are the gateway agents.

3.2 Creation of the Gateway Agent that Supports Interoperability Between an EX MAS and the Legacy One

An example of a simple gateway agent (GA) written in Java using the FIPA-OS toolkit[5] is demonstrated below. Basically, GA is composed of three main classes. In the first class, the constructor (line 1), GA sets a listener (line 5) and registers itself with the AMS of its platform (line 6). In line 7, GA calls the *registration_with_DF* method, where it registers itself with the DF along with all the available services provided by its internal agents.

In the second class (line 12), a method is created for each performative that is supported by GA. For instance, when GA receives a message, its listener set in the first class, calls *IdelTask* class and according to the incoming performative the appropriate method of *IdleTask* is executed e.g. code lines 14-18 handle any incoming *Request* performative. Therefore, when GA receives a *Request* from an external MAS its *handleRequest* method is executed (line 12) which in turn calls the *b_response* (third class).

When the third class (line 21) is initialized, GA validates the incoming *Request* message (line 26) and, if it is not understood it sends a *Not-understood* performative to the external sender agent, otherwise it sends an *Agree* performative. In the case where the message is understood, GA calls the *find_IntAgent* method (line 31) in order to find which of its internal agent hands the service indicated by the incoming message. In line 32, GA forwards the translated incoming message (line 26) to its appropriate internal agent along with the external agent's conversation ID. Once the internal agent has accomplished its task i.e. served the request, GA translates the results received by the former agent in order to generate an ACL message (line 35). Then, it replies to the external agent based on the conversation indicated by the conversation ID received from the internal agent (that GA had initially passed to it) via an *Inform* performative (line 36) which include the results.

Example of a simple Gateway Agent

```
1  public class GA extends FIPAOSAgent
2  {
3   public GA(String platform,String name,String ownership)
4    {
5     setListenerTask(new IdleTask());
6     registerWithAMS();
7     registration_with_DF();
8     ...
9    }
10  ...
11 }
12 public class IdleTask extends Task
13  {
14   public void handleRequest(Conversation conv)
15    {
16     ...
17     newTask(new b_response(conv),conv);
18    }
19   ...
20  }
21 public class b_response extends Task
22 {
23  protected void startTask()
24  {
25    //Translate the incoming ACL msg.
```

```
26   mes_transl=translate_mes_GA_to_IntA(ext_mes);
27    // If it is not OK send a Not-understood ACL msg,
28       otherwise send an Agree ACL msg.
29   ...
30    //If ACL msg is OK, serve the request.
31   internal_agent=find_IntAgent(service_req);
32   mes=contact_IntAgent(internal_agent,mes_transl,convID);
33   ...
34    //send results to the ext.agent via an Inform ACL msg
35   reply_mes=translate_mes_IntA_to_GA(mes);
36   response_mes(ext_agent,reply_mes,covID2);
37   ...
38  }
39 }
```

An example of serving a *Request* received by the gateway agent from an external agent is demonstrated in figure 3 with a message flow diagram.

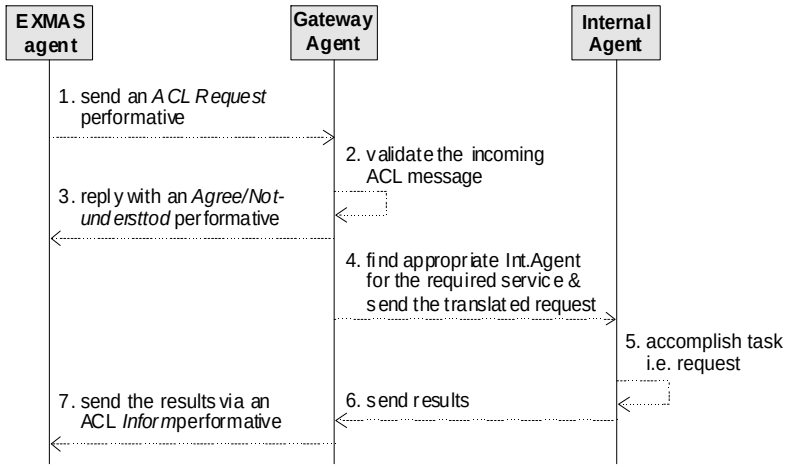

Fig. 3. Message flow between an external agent and the Gateway agent

3.3 Creation of the Gateway Agent that Supports Interoperability Between the Legacy MAS and an External One

The realization of this GA is completely upon the developer's point of view. According to the services required by its internal agents, GA should be programmed in order to know which external agent(s) provide the appropriate service(s), how to communicate with those agents and how to translate an internal agents' request to ACL messages and vice-

versa i.e. how the incoming ACL messages have to be translated to the form understood by its internal agents.

Note that amendments on the original architecture of the legacy MAS on which the FIPA-compliant gateways will be adopted, concerns only the agents that need to communicate with the gateway agent. This involves an extra *method* in the structure of each internal agent so as to enable them to send and receive a message from/to the appropriate gateway agent.

4 Testing the Interoperability of the FIPA-Compliant Gateways

Our research is based on the Synthetic Aperture Radar Atlas (SARA) active Digital Library[14], [15]. In order to achieve interoperability between our system and an external one, we have adopted the FIPA-compliant gateways approach. In the following section we give a brief discussion of SARA project, we demonstrate how interoperability can be achieved by using the approach outlined previously and we present results of experiment tests performed on the interoperability of our system.

4.1 The SARA Active Digital Library

SARA is an active digital library of multi-spectral remote sensing images of the earth from the SIR-C Shuttle mission, which provides web-based online access to a library of data objects at Caltech, the San Diego Supercomputer Center, and the University of Lecce in Italy. The objective of the SARA project is to develop an infrastructure for a high-speed, high-volume, multi-protocol distributed database, together with a means to attach distributed computing resources for data conversion, visualization and knowledge discovery[17].

A prototype MAS, which comprises both intelligent and mobile agents, has been developed to manage and analyse distributed multi-agency remote sensing data; more information can be found on our web-site[9]. The SARA architecture (figure 4) is composed of a collection of *information* and *web* servers, each of them having a group of agents, Local Interface Agents (LIA) and User Interface Agents (UIA) accordingly.

We separate mobile agents from stationary service agents. Our approach is to localize the most complex functionality in non-mobile LIAs, which remain at one location, providing resources and facilities to lightweight mobile agents that require less processor time to be serialized and therefore quicker to transmit. LIAs are stationary agents that provide an extensible set of services and a level of abstraction between resource servers and requesting mobile agents. UIAs provide a front end to the end user, for checking the user input and displaying the results.

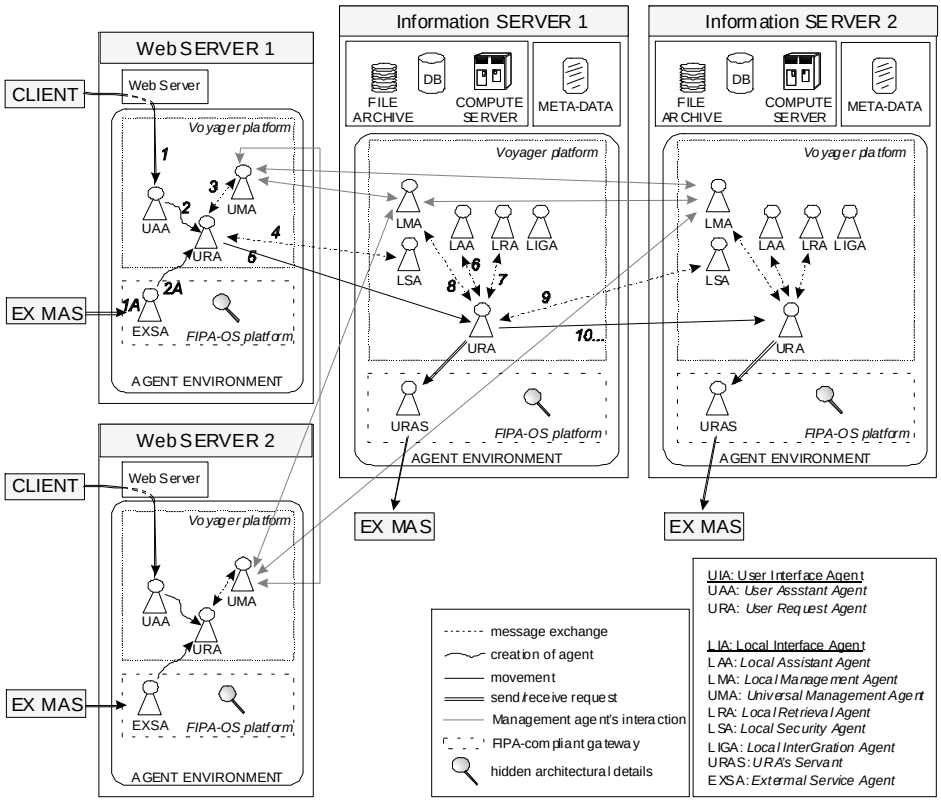

Fig. 4. The FIPA interoperable SARA architecture

4.2 SARA and FIPA Compliance

The introduction of FIPA interoperability into the SARA system enables it to communicate with other MAS and vice-versa. The union of SARA system with other MAS extends its capabilities by providing users with further information. For instance, information retrieved from the SARA system can be further enhanced by additional information gathered from a GIS system that is capable of interoperating with SARA. The longitude and latitude of a particular area of the earth can be used as parameters on a GIS (Geographic Information System) to retrieve land information such as street names, which can then be combined with the image based on geographical coordinates in SARA, resulting in a detailed map of the particular area. Likewise, an external MAS can interoperate with SARA and use its information.

The interoperability of the SARA system is based on the use of FIPA-compliant gateways which are implemented using FIPA-OS toolkit. The architecture of the SARA

system with added FIPA interoperability is depicted in figure 4. An external multi-agent system (EX MAS) can interoperate with SARA through the FIPA-compliant gateway (outlined by the dashed box) which is placed on every Web-server, where SARA can interoperate with an EX MAS through the FIPA-compliant gateway which is placed on every Information-server. The architecture of the FIPA-compliant gateways which is a slight variation of the architecture of FIPA-OS *configuration case 2*[6] is depicted in detail in figures 5.

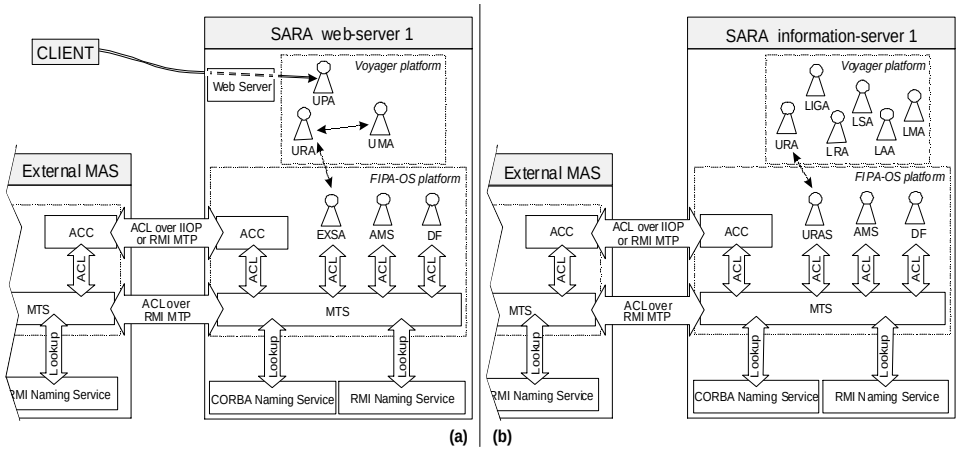

Fig. 5. Representation of the FIPA-compliant gateways: (a) on the Web-server and (b) on the Information-server

The EXSA agent is the *gateway* agent of the FIPA-compliant gateway placed on every Web-server. This agent is responsible for receiving a request from an external MAS i.e. a foreign agent, and passing it to the URA. EXSA can be considered *similar* to UAA; based on the fact that, as a client is represented by a UAA, an external MAS is represented by an EXSA. When URA receives the appropriate information from EXSA it processes its request (i.e. by starting its itinerary) as it would be instructed by a UAA agent. When URA finishes its job, it sends the results back to the EXSA which then passes this to the foreign agent from where the request has been initially placed. The resource access level and request's priority level is according to the EX MAS that accesses SARA.

The URAS agent is the *gateway* agent of the FIPA-compliant gateway placed on every Information-server. The purpose of this agent is to server URA with information gathered from external MAS. When URA needs to access an EX MAS, it passes its request to the URAS which is responsible of fulfilling URA's request. Ones, URAS has came in contact with the foreign agent of the appropriate EX MAS and has the results requested by the URA, it sends them back to URA. Until URAS has not acquired the results requested by

URA, URA is free to continue with its next task (if it has one), migrate to another information-server or wait for URAS agent's response.

4.3 Experiment Test Results

To test the interoperability of SARA system with an external one we have conducted our experiments using two different types of agent FIPA-compliant platforms. The first one was implemented using FIPA-OS toolkit (version 2_1_0-20030219000011, build:314) running on Unix and the second one was implemented on JADE framework[11] (version 2.4.1) running on Linux.

The tester agent of the FIPA-OS agent platform was a simple agent that was created to search the DF of SARA system for the EXSA's service and perform a Request. The second tester agent of the JADE agent platform was one of the tester agents of the Manchester node of Agentcities[1], which is running in the Agentcities test-bed since December 2001 hosted by UMIST (University of Manchester Institute of Science and Technology)[2].

The screenshots in figure 6 show the results of our experiments. The top picture is the console server of the SARA web server (running on Windows XP), the middle one is the console of the SARA information server (running on Unix) and the last one shows the execution of the tester agent of the FIPA-OS agent platform (running on Unix).

Initially, both of the tester agents performed a search on the DF of SARA to find the EXSA gateway agent's AID (Agent IDentifier) that provides the appropriate service. The interaction of an agent with the SARA DF is managed by FIPA-OS itself. Once, the tester agents have acquired the gateway agent's AID, they both sent a *Request* performative to EXSA, similar to the following one:

Example of a simple *Request* ACL message

```
(request
 :sender (agent_from_EX MAS)
 :receiver (EXSA)
 :content "coordinates 16.317 107.654 16.061 108.082 16.828
                       108.575 17.087 108.144"
 :language ASCII
 ...
)
```

The coordinates specified in the content of the ACL message corresponds to the collection of images required by the sender agent. When EXSA received the requests from the tester agents (figure 6a) it validated them. Since the incoming requests were valid, it replied to each of the tester agents with an *Agree* performative (figure 6c) and created for each of them a proxy of the URA agent locally. URA is the *internal* SARA agent that accepts as input Earth coordinates and gives as output a collection of images corresponding to the coordinates provided. The messages sent to each URA from EXSA

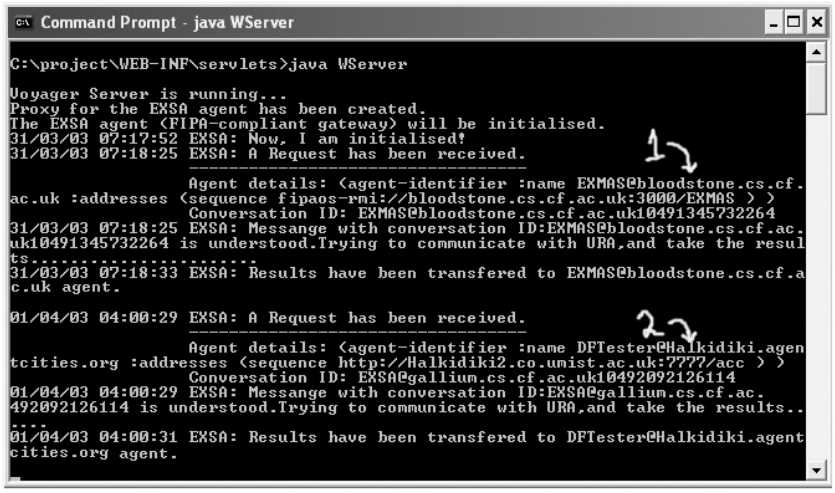

Fig. 6. Test results

were the tester agent's request translated into XML form (as understood by URA) and the conversation ID of the corresponding tester agent's interaction with EXSA.

After each URA has been initialized by EXSA, it communicated with its local management agent i.e. UMA, in order to receive the itinerary that has to follow through the SARA information servers in order to accomplish its task i.e. gather the information requested by EXSA. UMA is responsible for constructing every URA's itinerary in SARA according to the information provided by the latter and the current status of the system (known by UMA), i.e. availability of resources, server failures, number of agents on each server. UMA may also direct URA to collect the results of its query from a server which have already been stored by a previous agent having a similar query. The management agent's (UMA, LMA) interaction is described in [8].

For each URA, the steps of accomplishing their task may be traced by following the numbers on the diagram of the SARA architecture in figure 4 or with reference to the SARA information server's console in figure 6b. The console records the execution of every URA agent on the visited SARA information servers and reveals their interaction with the local stationary agents hosted by Voyager[16] agent platform.

After URA has accomplished its task, it sent the results back to the EXSA along with the conversation ID, initially received by EXSA, and self-terminated. Then EXSA replied to each of the tester agents based on the conversation indicated by the conversation ID received from its internal agent i.e. URA, via an *Inform* performative including a URL address (see figure 6b and 6c). The actual results could be acquired by accessing the corresponding URL address.

Details on the messages exchanged between the tester agents and the EXSA gateway agent, the translation of a *Request* ACL message performed by EXSA to the form understood by URA i.e. XML, and an example of results gathered by URA based on specific coordinates can be found in the Appendix.

5 Conclusion

In this paper we described how a developer may attach FIPA-compliant gateways to a legacy MAS. We discussed the advantages of adopting this approach and we demonstrated the successful interoperability provided by conducting experiment test on a MAS utilizing the FIPA-compliant gateways.

Currently we are developing a novel architecture for generic FIPA-compliant gateways that could be attached in a legacy MAS to provide automated FIPA interoperability with an external MAS. By the term automated we means that a developer would not need to have any knowledge of the FIPA specifications in order to make its system FIPA-compliant. Although, the proposed architecture of the generic FIPA-compliant gateways will support a limited number of performatives, a developer would be able to extend the

gateway agent class in order to support any performative that it will not be initially supported by the generic architecture.

References

1. AgentCities - a global, collaborative effort to construct an open network of on-line systems hosting diverse agent based services, http://www.agentcities.org (2003)
2. AgentCities node hosted by UMIST (University of Manchester Institute of Science and Technology), UK, http://www.agentcities.co.umist.ac.uk (2003)
3. Burg , B., Dale, J., Willmott, S.: Open Standards and Open Sources for Agent-Based Systems, Article in: Agentlink, news 6 (2001)
4. Charlton, P., Bonnefoy, D., Lhuillier, N., Gouaich, A.,Camenen, Y.: Dealing with interoperability for Agent Based Services, White paper, http://leap.crm-paris.com/agentcities/Resources/resources.html (2000)
5. FIPA-OS, http://www.nortelnetworks.com/products/announcements/fipa/index.html
6. FIPA-OS Inter-platform Communications Configuration Guide, http://www.nortelnetworks.com/products/announcements/fipa/index.html (2002)
7. Georgousopoulos, C., Rana, O. F.: An approach to conforming a MAS to a FIPA-compliant system. In First International Joint Conference on Autonomous Agents and Multi-Agent Systems - AAMAS 2002, ACM ISBN 1-58113-480-0, Italy, Bologna (2002) 968-975
8. Georgousopoulos, C., Rana, O. F.: Combining State and Model-based Approaches for Mobile Agent Load Balancing. In SAC 2003 - ACM Symposium on Applied Computing, ACM ISBN 1-58113-624-2, Melbourne, Florida, USA (2003) 878-885
9. http://www.cs.cf.ac.uk/Digital-Library/
10. http://www.fipa.org/docs/output/f-out-00065/
11. JADE (Java Agent DEvelopment Framework), http://sharon.cselt.it/projects/jade (2003)
12. Panti, M., Penserini, L., Spalazzi, L., Valenti, S.: A FIPA compliant agent platform for federated information systems. In ACIS International Journal of Computer & Information Science, volume 1, issue 3. Special issue on software engineering applied to networking & parallel/distributed computing, ISSN:1525-9293, USA (2000) 145-156
13. Poslad, S., Calisti, M.: Towards improved trust and security in FIPA agent platforms. Proceedings of Autonomous Agents 2000 Workshop on Deception, Fraud and Trust in Agent Societies, Spain (2000)
14. Yang, Y., Rana, O. F., Georgousopoulos, C., Walker, D. W., Williams, R., Aloisio, G.: Agent based data management in digital libraries. In Parallel Computing Journal, Elsevier Science, vol. 28, issue 5 (2002) 773-792
15. Yang, Y., Rana, O. F., Walker, D. W., Williams, R., Aloisio, G.: Towards an XML and Agent-Based Framework for the Distributed Management of Multi-Spectral Data. 6th International Digital Media Symposium, Bradford, UK (2001)
16. Voyager, Recursion Software, Inc., http://www.recursionsw.com/osi.asp (2003)
17. Williams, R.D., Sears, B.: A High-Performance Active Digital Library, Parallel Computing, Special issue on Metacomputing (1998)

APPENDIX: Test results

The messages received by the tester agent of the JADE agent platform:

```
(AGREE
 :sender (agent-identifier :name EXSA@gallium.cs.cf.ac.uk :addresses
 (sequence fipaos-rmi://gallium.cs.cf.ac.uk:3000/EXSA fipaos-rmi://gallium
 .cs.cf.ac.uk:3000/acc IOR:000000000000001149444c3a464950412f4d54533a312e30
 0000000000000001000000000000003000010000000000867616c6c69756d0004cf00000
 0000018afabcafe000000026dc432d3000000080000000000000000 iiop://gallium.cs.
 cf.ac.uk:4000/acc corbaname::gallium.cs.cf.ac.uk:4000/NameService#acc
                                          http://gallium.cs.cf.ac.uk:8080 ))
 :receiver (set ( agent-identifier :name DFTester@Halkidiki.agentcities.org
        :addresses (sequence http://Halkidiki2.co.umist.ac.uk:7777/acc )) )
 :content "coordinates 16.317 107.654 16.061 108.082 16.828 108.575 17.087
                       108.144"
 :language ASCII
 :conversation-id EXSA@gallium.cs.cf.ac.uk104920921126114
)

(INFORM
 :sender ( agent-identifier :name EXSA@gallium.cs.cf.ac.uk :addresses
 (sequence fipaos-rmi://gallium.cs.cf.ac.uk:3000/EXSA fipaos-rmi://gallium
 .cs.cf.ac.uk:3000/acc IOR:000000000000001149444c3a464950412f4d54533a312e30
 0000000000000001000000000000003000010000000000867616c6c69756d0004cf000000
 000018afabcafe000000026dc432d3000000080000000000000000 iiop://gallium.cs.
 cf.ac.uk:4000/acc corbaname::gallium.cs.cf.ac.uk:4000/NameService#acc
                                          http://gallium.cs.cf.ac.uk:8080 ))
 :receiver (set ( agent-identifier :name DFTester@Halkidiki.agentcities.org
        :addresses (sequence http://Halkidiki2.co.umist.ac.uk:7777/acc )) )
 :content "http://www.cs.cf.ac.uk/user/C.Georgousopoulos1/gallium/EXSA/0104
           03/gallium_8000_EXSA_1049209212627.xml"
 :language ASCII
 :conversation-id EXSA@gallium.cs.cf.ac.uk104920921126114
)
```

The message that is generated after the translation of the ACL message received by the EXSA gateway agent, into the XML form understood by its internal agent i.e. URA:

```
<?xml version='1.0'>
<!DOCTYPE message SYSTEM "message.dtd">
<Message type="request" id="CLIENTID">
<Context sender="//web_server1/EXSA" receiver="//web_server1/URA">
<Content> <querydef> &trackquery;</querydef> </Content>
</Message>
```

where the *trackquery* contains:

```
<?xml version='1.0'>
<!DOCTYPE trackquery SYSTEM "trackquery.dtd">
<trackquery>
```

```
<Condition><and>
<MoreThanOrEqual><left>latitude.upperleft</left><right>16.317</right>
</MoreThanOrEqual>
<MoreThanOrEqual><left>longtitude.upperleft</left><right>107.654</right>
</MoreThanOrEqual>
<MoreThanOrEqual><left>latitude.upperright</left><right>16.061</right>
</MoreThanOrEqual>
<LessThanOrEqual><left>longtitude.upperright</left><right>108.082</right>
</LessThanOrEqual>
<LessThanOrEqual><left>latitude.lowerleft</left><right>16.828</right>
</LessThanOrEqual>
<MoreThanOrEqual><left>longitude.lowerleft</left><right>108.575</right>
</MoreThanOrEqual>
<LessThanOrEqual><left>latitude.lowerright</left><right>17.087</right>
</LessThanOrEqual>
<LessThanOrEqual><left>longitude.lowerright</left><right>108.144</right>
</LessThanOrEqual>
</and></Condition>
</trackquery>
```

The results retrieved from the SARA DL based on the coordinates specified by the tester agent of the JADE agent platform are:

```
<?xml version="1.0"?>
<SARAMETADATA>
<SARATRACK IDTRACK="13106">
<NAME>Phnum Voeene, Cambodia</NAME>
<TRACKDATE>1994-04-16 00:00:00.0</TRACKDATE>
<WIDTH>4304</WIDTH>
<HEIGHT>7996</HEIGHT>
<CHANNELS>2</CHANNELS>
<SARACOORDS>
<SARACOORD><LAT>16.317</LAT><LON>107.654</LON></SARACOORD>
<SARACOORD><LAT>16.061</LAT><LON>108.082</LON></SARACOORD>
<SARACOORD><LAT>16.828</LAT><LON>108.575</LON></SARACOORD>
<SARACOORD><LAT>17.087</LAT><LON>108.144</LON></SARACOORD>
</SARACOORDS>
<SARAFILES>
<SARAFILE NAME="pr13106_byt_hh"><POLARIZATION>LHH</POLARIZATION></SARAFILE>
<SARAFILE NAME="pr13107_byt_hv"><POLARIZATION>CHV</POLARIZATION></SARAFILE>
</SARAFILES>
<SARASTORED><SERVER>server1</SERVER></SARASTORED>
</SARATRACK>
</SARAMETADATA>
```